1

ARMY RESEARCH LABORATORY

Distributed Heterogeneous Visualization, *Bop* and *Bop_View*

Jerry A. Clarke

ARL-CR-172

September 1994

prepared by

Computer Sciences Corporation
3160 Fairview Park Drive
Falls Church, VA 22042

under contract

DAAL03-89-7C-0088

NOTICES

Destroy this report when it is no longer needed. DO NOT return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
September 1994

3. REPORT TYPE AND DATES COVERED
Progress, September 1992 - October 1993

4. TITLE AND SUBTITLE
Distributed Heterogeneous Visualization *Bop* and *Bop_View*

5. FUNDING NUMBERS
C: DAAL03-89-7C-0088

6. AUTHOR(S)
Jerry A. Clarke

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Computer Sciences Corporation
3160 Fairview Park Drive
Falls Church, VA 22042

Army High Performance Computing
Research Center, University of Minnesota
1100 Washington Ave. South
Minneapolis, MN 55415

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
ATTN: AMSRL-OP-AP-L
Aberdeen Proving Ground, MD 21005-5066

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
ARL-CR-172

11. SUPPLEMENTARY NOTES
The Point of Contact for this report is Harold J. Breaux, U.S. Army Research Laboratory, ATTN: AMSRL-CI-A, Aberdeen Proving Ground, MD 21005-5067.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

With the increased use of parallel and super computers in scientific computing, the size of datasets that need to be visualized can easily reach into the gigabyte range. Even by utilizing data reduction techniques such as isosurface generation, scenes containing hundreds of thousand or millions of polygons are common. Standard techniques of data visualization quickly become overwhelmed and too time consuming to be practical.

New methods and utilities need to be developed to handle these massive datasets. *Bop* (Bag - O - Polygons), *Bop_View*, and associated utilities are an attempt to use distributed and parallel techniques to ease the processing of these datasets.

Bop is a data format designed for large number of polygons. A library of routines is provided for reading and writing this data to disk files. Additional routines allow this polygonal information to be shared across heterogeneous architectures. Finally, a application called *Bop_View* is provided to efficiently display the resulting information.

14. SUBJECT TERMS
visualization, distributed computing, computers, polygons

15. NUMBER OF PAGES
19

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

INTENTIONALLY LEFT BLANK.

LIST OF FIGURES

LIST OF TABLES

INTENTIONALLY LEFT BLANK.

1. INTRODUCTION

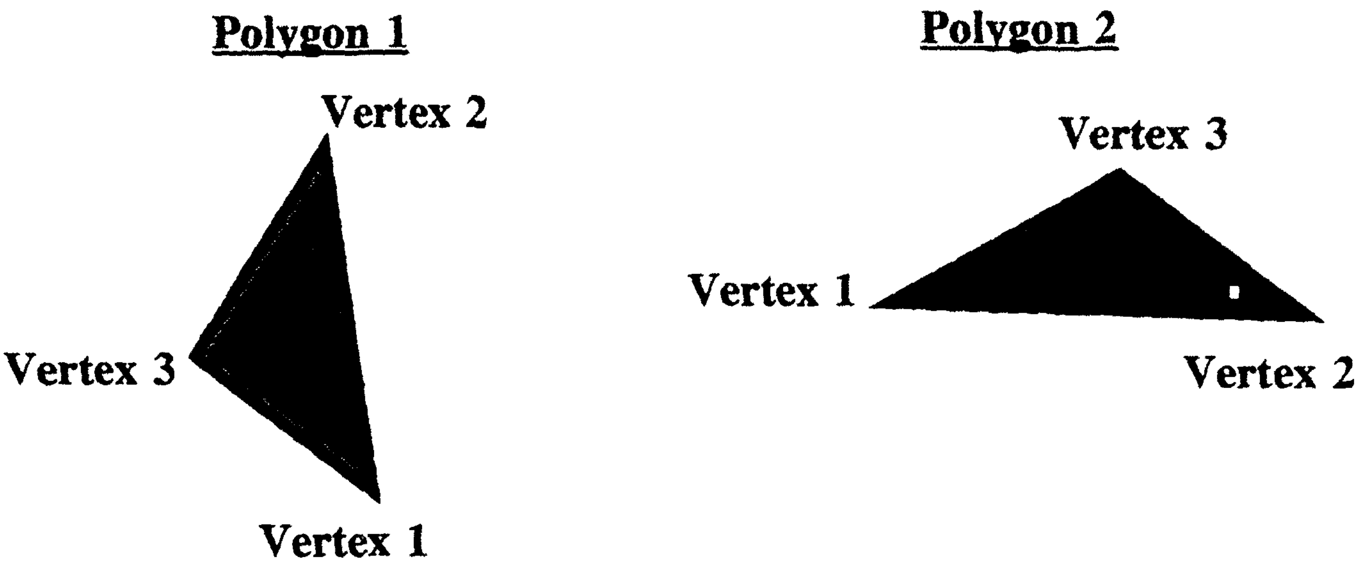
Three major items are described in this report: 1) the *Bop* data format, 2) subroutines for accessing disk files and networked polygons, and 3) *Bop_View*. *Bop_View is written using the subroutine interface.* The entire system is designed with the assumption that not all of the data may fit in physical memory. Therefore, there are options to handle this situation as it arises. Other utilities can be developed that utilize the networkability of the system and the simplicity of the data format interfaces. *Bop_p3d_cat* is an example of a utility that utilizes the subroutine interface to convert Plot_3D grids and solutions into a graphical format that can then be viewed with *Bop_View*. In a similar manner, a user may customize the system to deal with a specific data format or data location (data on a supercomputer, visualization on a workstation).

At the heart of the system is the *Bop* format. It is a simple, non-indexed binary polygon format. Basically, all of the information needed to render a polygon is contained within each polygon and a global header. By using a non-indexed format, all of the verticies need not be in memory at the same time. A header (actually at the end of the file) contains global minimum and maximum information about the entire polygon set. A *Bop* file is shown in Figure 1.

Each *Bop* file polygon reserves enough space for a global maximum number of verticies, even though it may not use them all. This global maximum is set at the time of file creation. This allows utilities to quickly move any polygon in the dataset via the Unix *seek* system call.

Bop files are not created or read directly, rather they are accessed through a set of library calls contained in *libbop.a*. The function contained in this library are as follows:

```
Bop_Ptr          *bop_open();
void             bop_close();
Bop_Polygon      *bop_read();
int              bop_write();
void             bop_clear();
int              bop_set();
```



```
int        number_of_vertices;
float      x, y, z, scalar;
float      x, y, z, scalar;                   Polygon 1
float      x, y, z, scalar;

int        number_of_verticies;
float      x, y, z, scalar;
float      x, y, z, scalar;                   Polygon  2
float      x, y, z, scalar;

long       total_verticies;
long       total_polygons;
float      xmin, ymin, zmin, scalar_min;      Global Information
float      xmax, ymax, zmax, scalar_max;
```

Figure 1. A *Bop* file.

bop_open() and *bop_close()* are used to access the disk files. A structure pointer containing necessary information for future access is returned by *bop_open()*. This structure pointer is then passed to all other functions. *bop_write()* appends polygons to the end of the *Bop* file while *bop_read()* returns an array of these polygons. *bop_clear()* is used to delete polygons from an existing file. *bop_set()* is used to set the global maximum number of verticies per polygon and to set the current read or write position. There is also a function, *bop_open_lock()*, which opens a file and also locks it using Unix file locking facilities. This is useful when several Unix processes need to access the same file. An example of using this library is given in the file *bop_test.c*. All routines and type declarations are declared in *bop.h*.

2. DISTRIBUTED PROCESSING

A library of communication routines known as *MRS* (Message Relay System) provides the basic connection between processes dealing with *Bop* information. *MRS* allows clients and servers to communicate across TCP/IP, shared memory, or Unix FIFO special file through a consistent abstraction. *libbop_mrs.a* contains routines that allow processes on the same processor or different architectures to send and receive polygon information and messages. eXternal Data Representation (XDR) is utilized to allow different internal binary formats to be accommodated. These routines are a superset of the *libbop.a* routines; this library can be used to read and write files as well as communicate between processes. Routines in *libbop_mrs.a* are as follows:

Bop_Ptr	*bop_mrs_open()
Bop_Ptr	*bop_open_file()
Bop_Ptr	*bop_open_tcp()
void	bop_mrs_close()
Bop_Polygon	*bop_mrs_read()
int	bop_mrs_write()
int	bop_mrs_set()
void	bop_mrs_clear()
int	bop_mrs_msg_send()
int	bop_mrs_msg_set()

The *libbop_mrs.2* routines are similar to the routines in *libbop.a* and are prototype in *bop_mrs.h*. These routines communicate on a structure known as a *Bop-O-Gram*. This sends polygons in packets of

BOP_O_GRAM_MAX_POLYS polygons (currently defined as 1000). This is the maximum polygons in a packet; if less are needed, less are sent.

In addition to polygon information, messages can be sent. *bop_mrs_msg_set()* takes the address of a dispatch routine to call when a message is received. *bop_mrs_msg_send()* is used to actually send the message. The message is a NULL terminated ASCII string and the meaning of the messages is application defined.

All messages and polygon packets sent through a connection established via *bop_mrs_open_tcp()* (the preferred interface) are transparently converted to XDR data. This allows architectures with different internal binary representations to efficiently share information. The TCP/IP conneciions do not use Remote Procedure Calls (RPC) and thus avoid the associated overhead.

3. BOP_VIEW: AN APPLICATION

Using the previously discussed subroutines, *Bop_View* was developed to aid in the visualization of *Bop* information. *Bop_View* is an X-window Motif application that allows polygonal information to be viewed on several different devices. Using "mixed mode" programming techniques, *Bop_View* will take advantage of SGI Graphics Language (GL) if it is available. Otherwise, the polygons are rendered to an X-window, SGI RGB file, a BRL-CAD pix file, or a Postscript file. (See Figure 2.) Because *Bop_View* utilizes XDR for network communications, networked polygons and commands need not originate from the same machine architecture. *Bop_View* currently executes on Silicon Graphics and Sun workstations.

Bop_View allows users to access files from disk or to wait for polygons to come across the network. Objects can be interactively rotated, translated, and scaled. The minimum and maximum cutoff for scalar values can be changed to highlight a selected range of interest. The image can also be rendered to a file in a number of different formats, and the *Bop* file can be saved to a local file. There are options that allow *Bop_View* to discard polygons once they are rendered; this allows an unlimited number of polygons to be rendered to the same scene.

Bop_View uses multiple command input streams for flexibility (Figure 3). The user can use the Graphical User Interface or send commands across the network. In this manner, *Bop_View* can be used interactively or from a Unix shell script. Polygons can be read from disk files or sent across the network

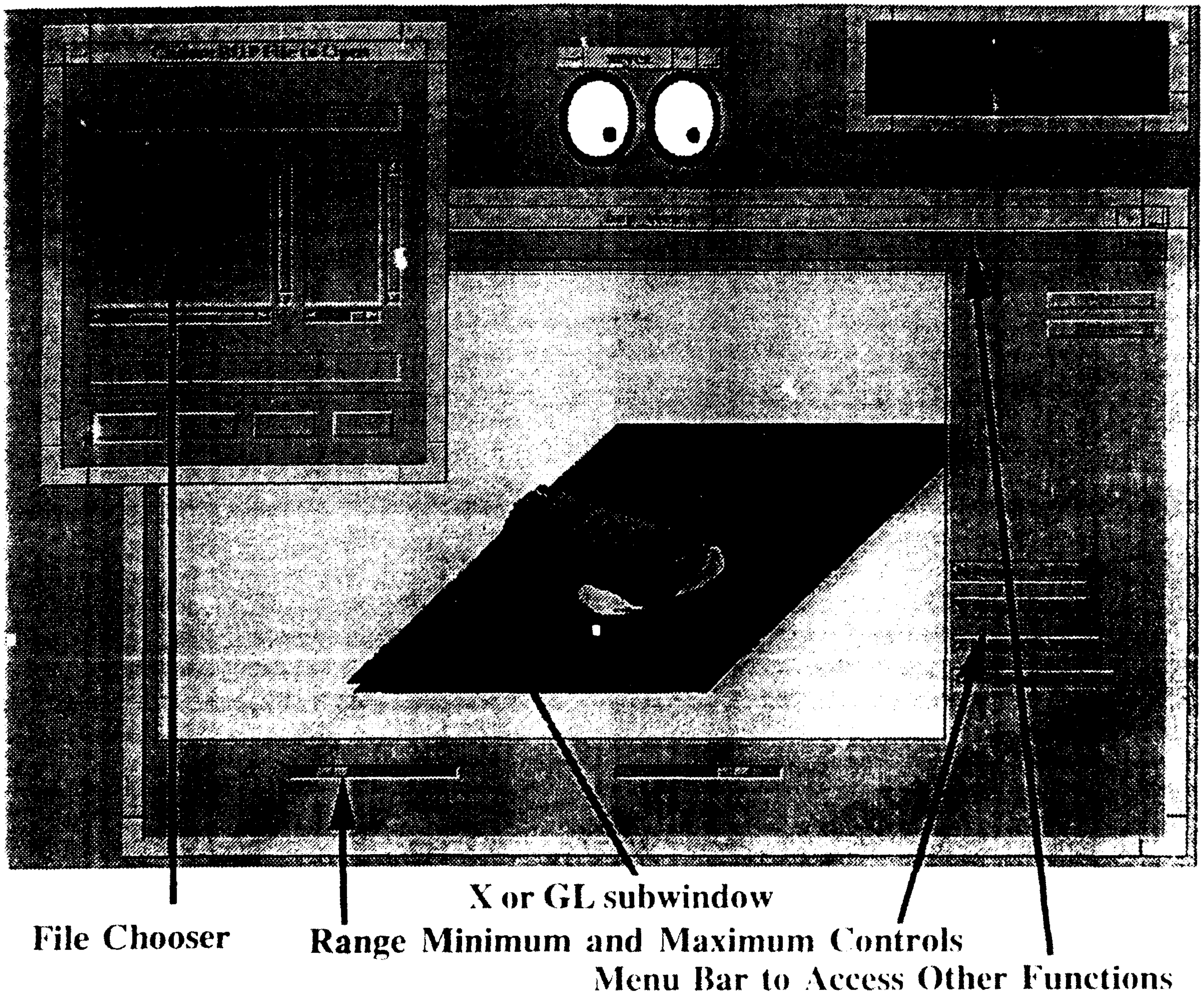

File Chooser
X or GL subwindow
Range Minimum and Maximum Controls
Menu Bar to Access Other Functions

Figure 2. [illegible] *View:* An application.

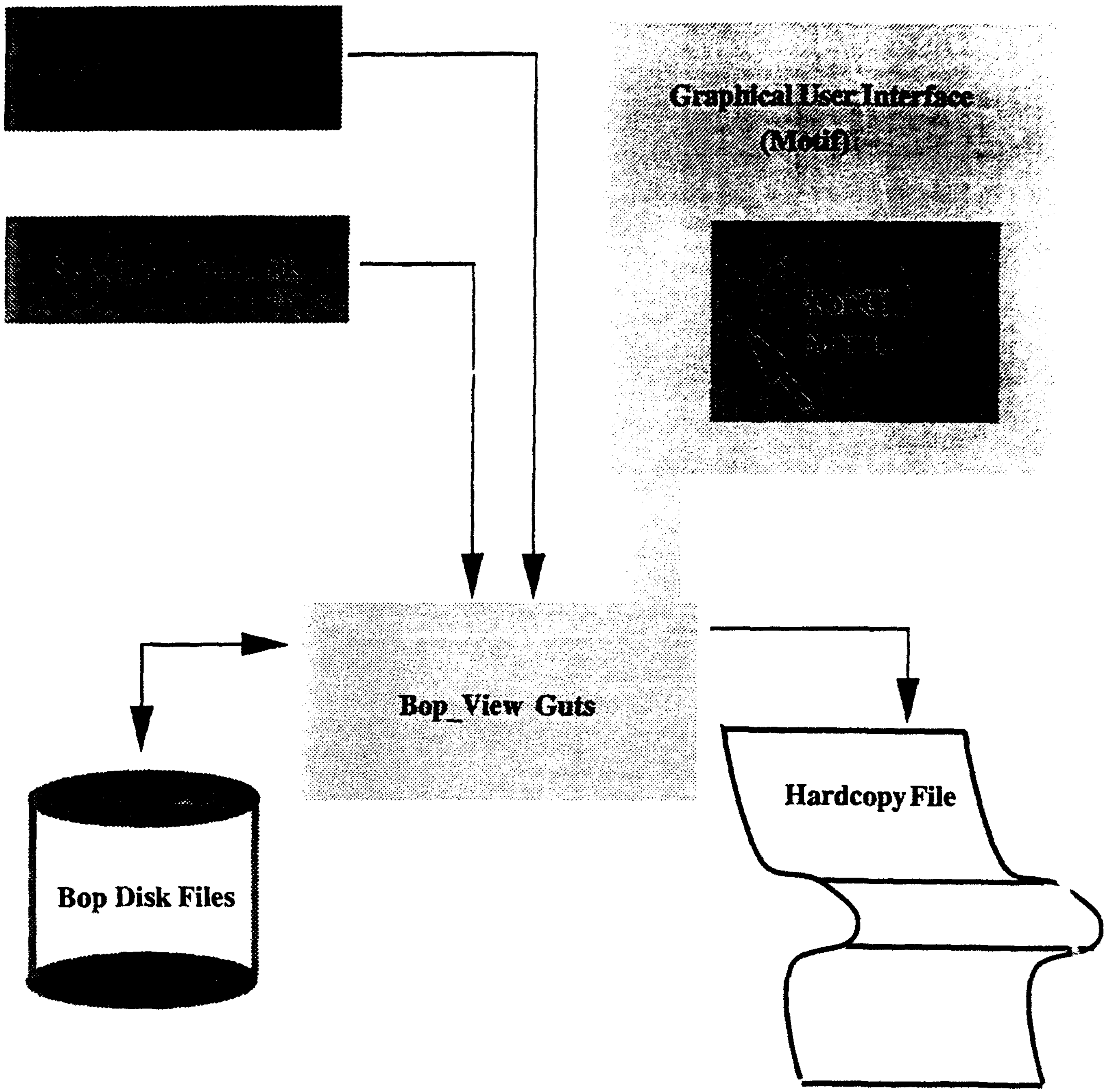


Figure 3. Multiple command input streams give *Bop View* flexibility.

and the current set of polygons can be saved to a disk file. In addition, graphical output can be directed to the X or GL subwindow and/or a hardcopy file. Formats for this hardcopy file include BRL-CAD pix, Silicon Graphics rgb, and color postscript.

To deal with large number of polygons, *Bop_View* allows the user to discard polygons after they are drawn. This allows thousands or millions of polygons to be rendered to a scene regardless of available physical memory. By utilizing composite Z buffer techniques, output is directed to the graphical subwindow and a hardcopy file.

BIG is a parallel isosurface generator (see BIG documentation) that runs on scalar, vector, and parallel machines such as the Kendall Square KSR-1. An interface to *BIG* has been developed that utilizes the *libbop_mrs* routines to output information directly to *Bop_View*. Huge datasets are processed on the KSR in parallel and the resulting polygons are received on the workstation. This information can be rendered to the screen, rendered to a file, and/or saved as a *Bop* file for later analysis.

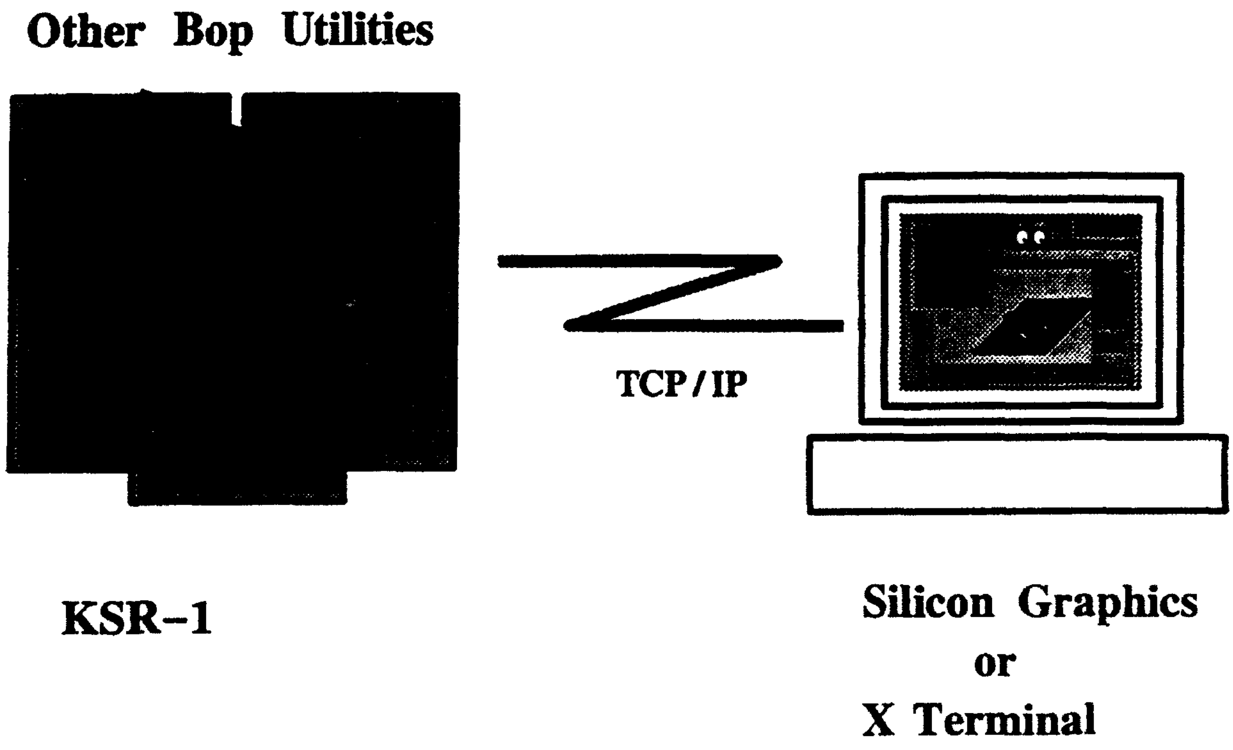


Figure 4. Typical *Bop View* application.

Some other utilities that aid in the use of the *Bop* format are as follows:

bop_stat — prints the header information from a *Bop* file.

bop_cat — puts *Bop* polygons into the network representation where they can be received by *Bop_View*.

bop_p3d_cat — converts a Plot_3D grid and solution into *Bop* network polygons where they can be received by *Bop_View*.

Commands may be issued through the GUI or by using the command:

```
bop_view_cmd   command_string   [options]
```

Table 1. *Bop_View* Usage

Command	GUI Menu Item	Effect
Ambient [0.0 – 1.0]	Preferences-Lighting-Ambient	Sets the ambient reflectance
Diffuse [0.0 – 1.0]	Preferences-Lighting-Diffuse	Set the diffuse reflectance
Draw	Redraw Button	Draws all currently saved polygons
Delete	File-Delete All	Clears all save polygons from memory
Exit	File-Exit	Exits
Light [0.0–1.0 0.0–1.0 0.0–1.0]	Preferences-Lighting-Direction	Sets the light source direction
Open	File-Open File	Reads in a Bop disk file
Passthru	Preferences-Other-Auto Print Passthru	Causes any incoming networked polygons to be rendered to hardcopy as well as graphical subwindow
Print	Print Button	Renders all saved polygons to hardcopy
Reverse	Edit-Reverse Normals	Reverses the normal vectors on all saved polygons.

Table 1. *Bop_View* Usage (continued)

Command	GUI Menu Item	Effect
Rotate [0.0 – 360 0 – 360 0 – 360]	Left Mouse Button in Subwindow	Sets rotation
Save filename	File-Save File	Writes all saved polygons to *Bop* disk file
Translate [x y z]	Middle Mouse Button in Subwindow	Sets translation
Scale [value]	Right Mouse Button in Subwindow	Sets scale factor
Update	Automatic	Updates GUI and subwindow
System command	none	Executes the command from inside *Bop_View*
Set Auto_Range [0 | 1]	Preferences-Other-Auto Range Update	If incoming polygons are outside the existing x, y, z or scalar range, the range is updated
Set Auto_Redraw [0 | 1]	Preferences-Other-Auto Redraw	If set off, the user must issue a "Draw" command. Useful for large number of polygons.
Set Auto_Save [0 | 1]	Preferences-Other-Auto Save Polygons	If set off, polygons are cleared from memory once they are rendered
Set Data_Range [min max]	Scalar Range Button or Color Data Sliders	Sets min and max for scalars
Set Show-Domain	Preferences-Other-Show Domain	Draws an outline of the current range
Set Light_On	Preferences-Lighting-Light On	Objects are lighted
Set Format [pix | sgi | ps]	Preferences-Other-Print File Format	Sets format for hardcopy
Set X_Range [min max]	X Range Button	Sets min and max for X
Set Y_Range [min max]	Y Range Button	Sets min and max for Y
Set Z_Range [min max]	Z Range Button	Sets min and max for Z

4. SUBROUTINES

void

bop_clear(Bop_Ptr *bp)

Deletes all of the polygons from a Bop file and resets the header information.

void

bop_close(Bop_Ptr *bp)

bop_mrs_close(Bop-Ptr *bp)

Closes a Bop file.

Bop_Ptr *

bop_open(char *filename)

bop_mrs_open_file(char *filename)

Creates a new Bop file or opens an existing file for appending.

Bop_Ptr *

bop_open_lock(char *filename)

Similar to bop_open except the file is also locked via fcnlt(2).

Bop_Polygon *

bop_read(Bop_Ptr *bp, int npoly)

bop_mrs_read(Bop_Ptr *bp, int npoly)

Reads up to npoly polygons from a Bop file. Returns a pointer to the first polygon or NULL on an error. Do not increment the pointer directly, rather use: BOP_NEXT_POLY(bp, poly_ptr). The space for these polygons is allocated via calloc(). The application is responsible for freeing this space.

int

bop_set(Bop_Ptr *bp, int what, int value)

bop_mrs_set(Bop_Ptr *bp, int what, int value)

Sets state of a Bop file. Valid values for "what" are BOP_CUR_POLY OR BOP_VPP (verts per polygon). Setting BOP_CUR_POLY positions the Bop file to that polygon (zero based) while setting BOP_VPP sets the maximum verticies per polygon. This may only be set before any polygons have been written to the Bop file.

int
bop_write(Bop_Polygon *bpoly, int npoly, Bop_Ptr *bp)

Writes npoly polygons pointed to by bpoly to the Bop file. Use BOP_NEW_POLY(bp, npoly) to allocate space for new polygons.

Bop_Ptr *
bop_mrs_open_tcp(char *hostname, int port_num)

Opens a TCP/IP connection on port_num. If port_num is zero, a unique number is generated using the user's UID; this is the preferred method.

Bop_Ptr *
bop_mrs_open(MRS_NODE *node)

Opens a connection on an existing MRS node. This allows the user to change the defaults of the connection such as size and location of data buffer. This is not recommended without a detailed knowledge of MRS.

void
bop_mrs_msg_call(Bop_Ptr *bp, char *data)

Sends the NULL terminated string as a message to the connection described by *bp.

int
bop_mrs_msg_set(Bop_Ptr *bp, void (*msg_routine)())

Sets the subroutine to call when bop_mrs_read() receives a message instead of polygon information. The subroutine is called with a char pointer that points to the string which passed to bop_mrs_msg_send().

```
#include <bop_mrs.h>
/* Write 2 triangles as a Bop-O-Gram */
main(argc, argv)
int                         argc;
char                        *argv[];
{
int                         i, j, n_triangles = 2;
float                       xstart = 0.0, ystart = 0.0, zstart = 0.0;
double                      atof();
Bop_Polygon                 *bpoly, *bpoly_start; /* Polygons */
Bop_Ptr                     *bp; /* Bop_file Pointer */

if(argc < 2){
                    fprintf(stderr, "Usage:  %s hostname\n", argv[0]);
                    exit(0);
                    }
if(argc > 2){
                    xstart = atof(argv[2]);
                    ystart = atof(argv[3]);
                    zstart = atof(argv[4]);
                    }

fprint(stderr, "Connecting to %s\n", argv[1]);
bp = bop_mrs_open_tcp(argv[1], 0); /* Choose port # based on UID */
bop_mrs_set(bp, BOP_VPP, 3); /* Set Verts/Poly for new files*/
bpoly_start = bpoly = BOP_NEW_POLY(bp, n_triangles);/* Allocate New Polys */
for(i=0; i < n_triangles; i++){
                    bpoly->nvert = 3;
                    bpoly->vert[0].x = xstart + i;
                    bpoly->vert[0].y = ystart + 0.0;
                    bpoly->vert[0].z = zstart +i;                 /* Data for vertex 1 */
                    bpoly->vert[0].data = 10.0 * i;
```

```
                        bpoly->vert[1].x = xstart + i + 1;
                        bpoly->vert[1].y = ystart + 0.0;
                        bpoly->vert[1].z = zstart + i;                      /* Data for vertex 2 */
                        bpoly->vert[1].data = 10.0 * (i + 1.0);

                        bpoly->vert[2].x = xstart + i;
                        bpoly->vert[2].y = ystart + 1.0;
                        bpoly->vert[2].z = zstart +i;                       /* Data for vertex 3 */
                        bpoly->vert[2].data = 10.0 * (i + 2.0);

                        bpoly = BOP_NEXT_POLY(bp, bpoly);
                        }
        bop_mrs_write(bpoly_start, n_triangles, bp);   /* Ship it!! */
        bop_mrs_close(bp);   /* Close connections */
        }
```

INTENTIONALLY LEFT BLANK.

5. REFERENCES

DDN Network Information Center. XDR: External Data Representation Standard, RFC-1014. Menlo Park, CA, June 1987.

Dykstra, P. C. "The BRL-CAD Package, An Overview." Ballistic Research Laboratory, Aberdeen Proving Ground, MD, October 1988.

Moss, G. S. "The 'lgt' Lighting Model." Ballistic Research Laboratory, Aberdeen Proving Ground, MD, October 1988.

Muus, M. J. "Workstations, Networking, Distributed Graphics, and Parallel Processing." Ballistic Research Laboratory, Aberdeen Proving Ground, MD, October 1988.

INTENTIONALLY LEFT BLANK.

No. of
Copies Organization

2 Administrator
Defense Technical Info Center
ATTN: DTIC-DDA
Cameron Station
Alexandria, VA 22304-6145

1 Commander
U.S. Army Materiel Command
ATTN: AMCAM
5001 Eisenhower Ave.
Alexandria, VA 22333-0001

1 Director
U.S. Army Research Laboratory
ATTN: AMSRL-OP-SD-TA,
Records Management
2800 Powder Mill Rd.
Adelphi, MD 20783-1145

3 Director
U.S. Army Research Laboratory
ATTN: AMSRL-OP-SD-TL,
Technical Library
2800 Powder Mill Rd.
Adelphi, MD 20783-1145

1 Director
U.S. Army Research Laboratory
ATTN: AMSRL-OP-SD-TP,
Technical Publishing Branch
2800 Powder Mill Rd.
Adelphi, MD 20783-1145

2 Commander
U.S. Army Armament Research,
Development, and Engineering Center
ATTN: SMCAR-TDC
Picatinny Arsenal, NJ 07806-5000

1 Director
Benet Weapons Laboratory
U.S. Army Armament Research,
Development, and Engineering Center
ATTN: SMCAR-CCB-TL
Watervliet, NY 12189-4050

1 Director
U.S. Army Advanced Systems Research
and Analysis Office (ATCOM)
ATTN: AMSAT-R-NR, M/S 219-1
Ames Research Center
Moffett Field, CA 94035-1000

No. of
Copies Organization

1 Commander
U.S. Army Missile Command
ATTN: AMSMI-RD-CS-R (DOC)
Redstone Arsenal, AL 35898-5010

1 Commander
U.S. Army Tank-Automotive Command
ATTN: AMSTA-JSK (Armor Eng. Br.)
Warren, MI 48397-5000

1 Director
U.S. Army TRADOC Analysis Command
ATTN: ATRC-WSR
White Sands Missile Range, NM 88002-5502

1 Commandant
U.S. Army Infantry School
ATTN: ATSH-WCB-O
Fort Benning, GA 31905-5000

Aberdeen Proving Ground

2 Dir, USAMSAA
ATTN: AMXSY-D
AMXSY-MP, H. Cohen

1 Cdr, USATECOM
ATTN: AMSTE-TC

1 Dir, USAERDEC
ATTN: SCBRD-RT

1 Cdr, USACBDCOM
ATTN: AMSCB-CII

1 Dir, USARL
ATTN: AMSRL-SL-I

5 Dir, USARL
ATTN: AMSRL-OP-AP-L

No. of
Copies Organization

1 Computer Sciences Corporation
ATTN: Dr. David Brown
3160 Fairview Park Dr.
Mail Code 265
Falls Church, VA 22042

Aberdeen Proving Ground

11 Dir, USARL
ATTN: AMSRL-CI, William Mermagen
AMSRL-CI-A, Harold Breaux
AMSRL-CI-AC,
John Grosh
Phillip Dykstra
Jerry Clarke
Deborah Thompson
Jennifer Hare
Eric Mark
Richard Angelini
Kathy Burke
AMSRL-CI-C, Walter Sturek

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number ARL-CR-172 Date of Report September 1994

2. Date Report Received

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.)

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.)

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate.

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.)

CURRENT ADDRESS

Organization

Name

Street or P.O. Box No.

City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD ADDRESS

Organization

Name

Street or P.O. Box No.

City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)

(DO NOT STAPLE)

DEPARTMENT OF THE ARMY

OFFICIAL BUSINESS

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 0001, APG, MD

Postage will be paid by addressee

Director
U.S. Army Research Laboratory
ATTN: AMSRL-OP-AP-L
Aberdeen Proving Ground, MD 21005-5066